ROBUST AND RECONFIGURABLE VERIFICATION PLATFORM FOR NOC IN IOT SOC DESIGN

Tarun Kumar Nagori[1], Dr. Vasundara Patel K S[2]
[1]M. Tech., [2]Associate Professor,
VLSI Design and Embedded System,
BMSCE, Bengaluru,India

Abstract- IoT (Internet of Things) is growing at a rapid rate and will be connected with the humans in the near future. IoT design has many applications which make it beneficial to verify it effectively. But, design verification has become very difficult today as the integration has gone to a new high to make a SoC (System on chip). So this paper deals with a NoC (Network on chip) on SoC for IoT design verification. To overcome building verification environment every time, a methodology is used in this paper which built a verification environment for a particular application. Whenever a similar/same application design comes for verification, with less or no modification, the verification environment can be automated using Perl language to generate a desired verification environment as per the application design requirements. At the end, it will be concluded that using this methodology, one can build the verification environment once and then can generate it automatically as per the specifications.

Keywords— IoT; NoC; SoC; AMBA AHB; Perl

I. INTRODUCTION

The rapid success of VLSI technology has enabled the integration of millions of transistors into a lone chip to make a SoC (System-on-Chip). As the integration and the number of masters like processor, etc. have increased, it has given rise to the interconnect issues, which makes verification of the particular design difficult task to accomplish. Nearly, 75% of the design cycle time is spent on functional verification.

To avoid this issue or the problem, NoC (Network on-chip) architecture is proposed, communicating with the design. It outlines the services, infrastructure and the power consumption and utilization of the resources.

This paper deals with faster IOTs through NoC level communication structure. It also deals with automating the test bench design using Perl. The network interface is AMBA's AHB (Advanced High-performance Bus).

This paper contains;-

1) Verification of AHB protocol using Verilog and generate coverage reports on Questa-Sim.
2) Test bench for any number of master/slaves is generated through Perl.

Simulation waveforms and coverage reports of the design by passing test cases have been shown in this paper.

Objective of the study:-

1) To study AMBA AHB specifications.
2) To generate the test bench design considering automation at the end to be used.
3) To generate the Test plan comprising Test cases implementation.
4) Automate the test bench design for any number of master/slave combinations using Perl.

II. AMBA AHB PROTOCOL

AMBA AHB is a high-performance, high clock frequency bus architecture. It enables the communication of processors, on-chip memories and off-chip external memory interfaces with low-power peripheral devices [1]. AHB is used to address the requirements of synthesizable designs needing high performance. It supports multiple bus masters and operates on high-bandwidth [3].

Some of the features include:-

a) single-cycle bus master handover
b) burst transfers
c) split transactions
d) non-tristate implementation
e) single-clock edge operation
f) wider data bus configurations

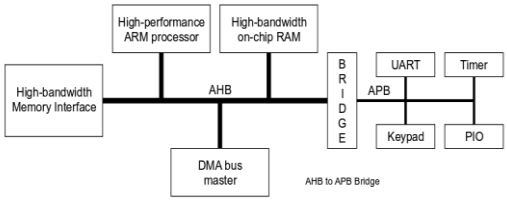

Fig 1:- AMBA Architecture

III. PROPOSED METHODOLOGY

Initially, the I/O ports of the design with their widths are specified in ".xls" file and are extracted using Perl and a module is generated in a Verilog file. Then, a test bench is designed for a particular master/slave number such that it works without errors for any combination of master/slave. RTL (Register Transfer Level) design is verified with the test bench having same number of master/slave by passing specific test cases.

At last, the test bench is automated using Perl to generate the required test bench comprising the same number of master/slave combinations as in the RTL design. Different test cases are passed at the runtime to different masters/slaves for generating the simulation waveforms. Finally, Coverage reports of the design are generated.

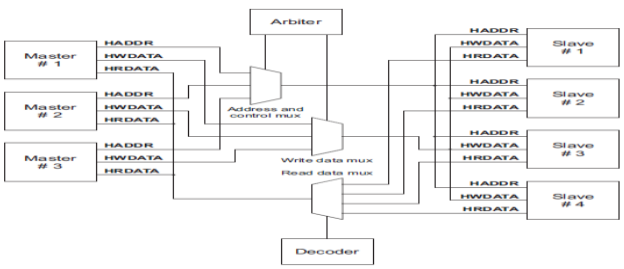

Fig 2:- AHB Interconnection

IV. VERIFICATION ENVIRONMENT

Purpose of the verification environment is to generate stimulus, apply it to the DUT (Design under test) and check the result to verify the functionality [2]. Later, test cases may be modified based on the coverage reports. Above figure shows the connection between the master and slave in AHB. The verification environment includes the DUT, test bench, memory to store data and test cases passed. This environment can be divided into three parts, DUT, which is the Verilog design of the application, test program/test bench including test cases, and the top module which connects other two parts simultaneously.

V. SIMULATION WAVEFORMS

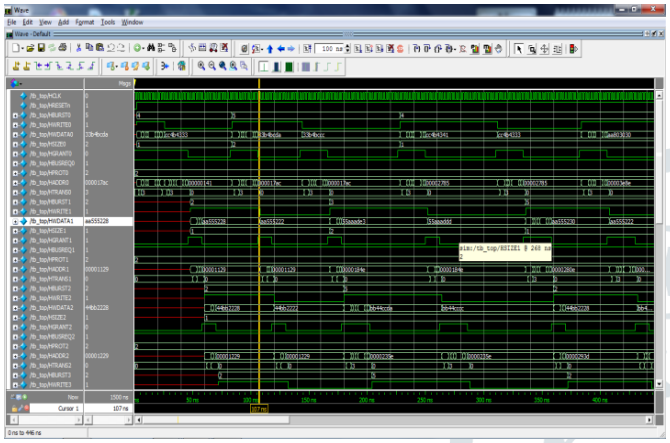

Fig 3:- 4 masters operation

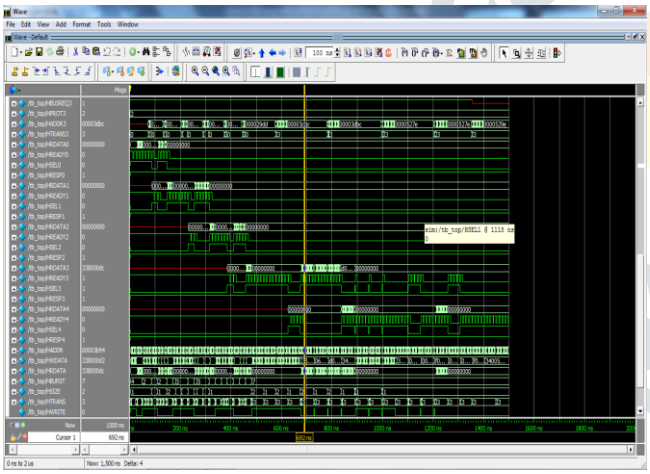

Fig 4:- 5 slaves operation

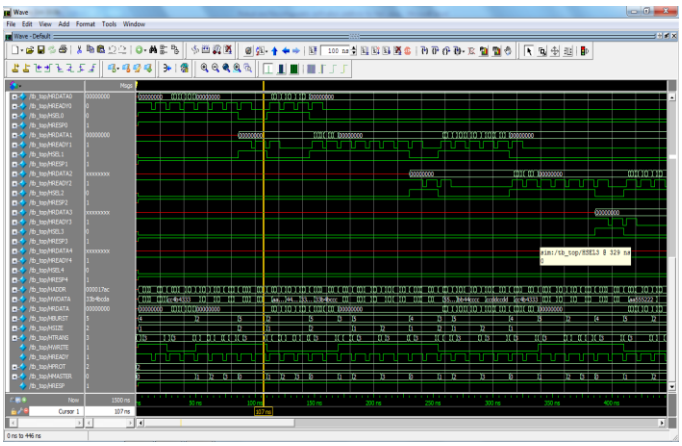

Fig 5:- Output MUX data

VI. CODE COVERAGE REPORTS

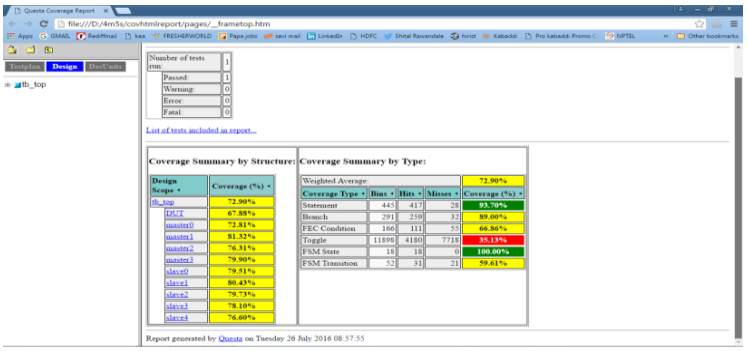

Fig 6:- coverage report in HTML

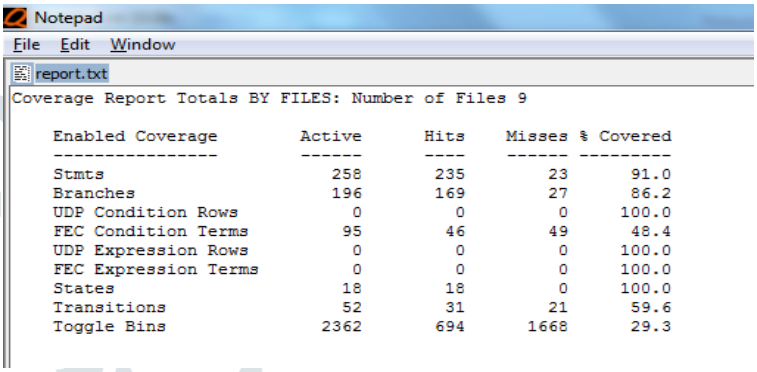

Fig 7:- coverage report in text

VII. CONCLUSION AND FUTURE WORK

The methodology used in this paper represents an environment generating the automated test bench design for different master/slave combinations using Perl. It can be concluded that for the same/similar IoT applications, verification will be done quickly with less or no modifications. This methodology will help in "time to market". The above simulation waveforms and coverage reports are for 4 masters and 5 slaves operation, which have a range from 1 to 16 each. Another verification module/test bench can be generated by changing master and slave number and its waveforms and results can be studied within small time. At the end, we can say that this paper deals with scripts, designing test benches and verifying them.

As far as the future work is concerned, this project will be done in System-Verilog and UVM methodologies for better simulation analysis and results.

VIII. ACKNOWLEDGEMENT

We, the authors of this paper would like to express our gratitude and sincere thanks to Mr. Prabhu Bhairi from Sevitech Systems, for his valuable guidance, advices and support for this project. We thank him for his constant encouragement and support for enabling us and also the Center of Excellence (CoE) lab, BMSCE for providing the required tools for this project.

REFERENCES

[1] "AMBA AHB"- Specification by ARM Limited.
[2] Questa SIM User's Manual, by Mentor Graphics.
[3] https://en.wikipedia.org/wiki/Advanced_Microcontroller_Bus_Architecture
[4] Saurin Shah, Nirav Patel, "Systemverilog Based AMBA AHB Verification Environment", International Journal of Engineering Research & Technology (IJERT), Vol. 3 Issue 5, May – 2014.